

# Respaldos multinivel robustos y simples utilizando rsync

Seminario Admin-UNAM  
8 de diciembre, 2005

Gunnar Wolf - [gwolf@gwolf.org](mailto:gwolf@gwolf.org)  
Instituto de Investigaciones Económicas, UNAM  
Desarrollador del Proyecto Debian

[http://www.gwolf.org/seguridad/respaldos\\_con\\_rsync/](http://www.gwolf.org/seguridad/respaldos_con_rsync/)

# Respaldos multinivel robustos y simples utilizando rsync

1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra
2. El uso de las ligas en los sistemas de archivos en Unix
3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)
4. El sistema rsync
5. Implementación simple de un esquema de respaldos multinivel

# 1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra

¿Cómo hacemos nuestros respaldos?

---

## Un esquema muy poderoso, muy profesional...

- **Cruzo los dedos y espero que no pase nada**
  - Nunca me ha pasado nada desde que me corrieron del trabajo anterior
  - Ya aprendí, y ahora soy un buen administrador - ¡Soy invulnerable!
  - ¡Chin! Ya me volvieron a despedir :-/
  
- **Realizo respaldos periódicos utilizando *fulanator***
  - Nunca he tenido que probar los mecanismos de recuperación
  - Funciona de maravilla en la plataforma que uso actualmente
  - Tiene una interfaz muy bonita

# 1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra

## Respaldos en cinta

---

La cinta es el medio favorito para respaldos masivos

- Es compacta, puedo guardar terabytes en un cajón
  
- Es relativamente barata
  - Unidades caras, medio físico medianamente accesible (del orden de 100 pesos por cinta, para decenas de GB)
  
- Si necesito mayor volúmen, puedo comprar un robot multicintas.
  - Tremendamente caros y voluminosos

# 1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra

## Respaldos en cinta

---

Sin embargo, no puedo recomendar su uso más que como un medio auxiliar en un esquema de respaldos más completo

- La recuperación de archivos es muy lenta
  - La cinta es un medio de acceso secuencial
  - Requiere el uso de un programa específico, no es *montable* desde el sistema
  
- Las cintas tienen una vida útil corta
  - En el mejor de los casos, 10 usos
  - Aún eso es demasiado arriesgado
  
- El fallo de un bit en el medio magnético puede destruir el respaldo entero
  - Muchas veces usamos compresión para aprovechar mejor el espacio
  - Si usamos paquetería de respaldos, muchas veces no podemos evitar el uso de compresión o el algoritmo a emplear

# 1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra

## Discos ópticos

---

Mucha gente ha preferido el uso de medios ópticos (CD, DVD) para realizar respaldos

- **Tanto medio como unidades muy baratos y confiables**
  - Aunque de baja capacidad - 700MB, 4GB, 9GB en el mejor de los casos (mas compresión)
- **Medio rápido y fácil de comprobar para la recuperación**
  - Sin compresión, uso el disco como una unidad más del sistema
  - Incluso puedo usar discos con compresión directamente desde algunos sistemas operativos
- **El medio reescribible es muy duradero**
  - El proceso de grabación nos reporta si hay defectos importantes en el medio
  - Aunque sensiblemente más lento
- **Existen robots cambiadores**
  - En mi experiencia, no son muy confiables

## 1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra

Principal problema con el uso de medios removibles

---

Los medios removibles son susceptibles a un fallo muy común:

# Híiiiiiiiijole...

¡Se me olvidó meter (la cinta|el disco) de hoy!

¡Yo creí que sí estaba haciendo el respaldo, pero el cable SCSI está zafado desde hace meses!

(y yo de tarado que no leo las bitácoras)

# 1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra

## Respaldos a disco

---

Confiando en la capacidad de recordar sus tareas a tiempo, le encargo a mi servidor el llevar a cabo periódicamente los respaldos a disco

- El espacio en disco es relativamente barato
  - Vale más mi tranquilidad que un disco duro adicional de 400GB
- Al hacer los respaldos en horarios de menor actividad, no sobrecargo la red en horas hábiles
- Es *muy* fácil verificar los respaldos y recuperar la información

# 1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra

## Respaldos a disco

---

...Pero tampoco estos son todo lo que necesitamos

- **¡Me cepillé el respaldo *junto con* los datos!**
  - Claro está, ninguna máquina debe ser destinatario de *su propio* respaldo
- **No tenemos protección ante desastres mayores**
  - Se incendió el cuarto de servidores, y...
  - Ahora bien... De los que hacen respaldo a cinta o medios ópticos, ¿quién los guarda a más de 50 metros del servidor?
- **No tengo un respaldo histórico**
  - Fundamental para datos sujetos a auditoría
- **Un intruso que penetra a un servidor tiene acceso a los datos de los demás**
  - NO USEMOS el esquema de "respaldo de A en B, respaldo de B en A"
  - Debemos designar un host específico para los respaldos
    - ▶ Aislado de la red externa (y, de ser posible, de la red interna)
    - ▶ Físicamente alejado (recuerden los incendios/temblores)
    - ▶ Consideremos el manejo de cifrado

## 1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra

No existe un claro ganador

---

# No hay un esquema perfecto

Debemos combinar estas soluciones según nos resulte más conveniente para nuestras necesidades específicas

## Respaldos multinivel robustos y simples utilizando rsync

1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra
- 2. El uso de las ligas en los sistemas de archivos en Unix**
3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)
4. El sistema rsync
5. Implementación simple de un esquema de respaldos multinivel

## 2. El uso de las ligas en los sistemas de archivos en Unix

Inodos, ligas... ¿Multipresencia?

---

En todos los sistemas de archivo nativos a cualquier sistema tipo Unix:

- Los directorios incluyen únicamente *apuntadores* a los datos
  
- Cada archivo se guarda en un *inodo*
  - La localización física de un conjunto de datos en una partición
  
- Cada inodo puede tener más de un nombre, y estar en más de un punto del árbol de directorios
  - Esto se llama una *liga dura* (detalles respecto las ligas más adelante)
  - Los nodos subsecuentes no apuntan al nombre original, sino que directamente a los datos en cuestión
  - Podemos eliminar a cualquiera de estos (incluyendo al original), y los demás se mantienen sin modificación
  - Claro está, sólo puedo crear ligas duras a archivos que estén dentro de la misma *partición*

En Unix tenemos más de un tipo de ligas - es importante comprenderlas para poder avanzar al siguiente punto

#### ▪ Ligas duras

- Las descritas anteriormente
- Cada uno de los archivos es equivalente, no hay uno maestro
- Deben estar en la misma partición
- Poco utilizadas en la administración diaria
- No podemos hacer ligas duras a directorios
  - ▶ Esto podría crear ciclos en la estructura del árbol

#### ▪ Ligas simbólicas

- Una entrada en el directorio que apunta a otra entrada del directorio
- Si elimino a la copia maestra, elimino la información
  - ▶ y todas las ligas a ella quedan rotas
- Son hasta cierto punto equivalentes a los *accesos directos* en Windows

## 2. El uso de las ligas en los sistemas de archivos en Unix

### Ligas duras - Un ejemplo

---

```
gwolf@lactop:/tmp$ cat > mis_datos.txt
Estos son mis datos, y los quiero. Me son muy importantes, los quiero mucho.
gwolf@lactop:/tmp$ ls -l mis_datos.txt
-rw-r--r-- 1 gwolf gwolf 77 2005-12-08 12:52 mis_datos.txt

gwolf@lactop:/tmp$ ln mis_datos.txt tambien_aqui.txt
# equivalente a cp -l mis_datos.txt tambien_aqui.txt
gwolf@lactop:/tmp$ ls -l mis_datos.txt tambien_aqui.txt
-rw-r--r-- 2 gwolf gwolf 77 2005-12-08 12:52 mis_datos.txt
-rw-r--r-- 2 gwolf gwolf 77 2005-12-08 12:52 tambien_aqui.txt

gwolf@lactop:/tmp$ cat >> tambien_aqui.txt
Mis datos crecen!
gwolf@lactop:/tmp$ cat mis_datos.txt
Estos son mis datos, y los quiero. Me son muy importantes, los quiero mucho.
Mis datos crecen!

gwolf@lactop:/tmp$ rm mis_datos.txt
gwolf@lactop:/tmp$ ls -l tambien_aqui.txt
-rw-r--r-- 1 gwolf gwolf 95 2005-12-08 12:54 tambien_aqui.txt
gwolf@lactop:/tmp$ cat tambien_aqui.txt
Estos son mis datos, y los quiero. Me son muy importantes, los quiero mucho.
Mis datos crecen!
```

## 2. El uso de las ligas en los sistemas de archivos en Unix

### Ligas simbólicas - Un ejemplo

---

```
gwolf@lactop:/tmp$ cat > mis_datos.txt
Estos son otros datos, y aunque no parezca, los quiero también.
gwolf@lactop:/tmp$ ls -l mis_datos.txt
-rw-r--r-- 1 gwolf gwolf 64 2005-12-08 13:04 mis_datos.txt

gwolf@lactop:/tmp$ ln -s mis_datos.txt tambien_aqui.txt
gwolf@lactop:/tmp$ ls -l mis_datos.txt tambien_aqui.txt
-rw-r--r-- 1 gwolf gwolf 64 2005-12-08 13:04 mis_datos.txt
lrwxrwxrwx 1 gwolf gwolf 13 2005-12-08 13:05 tambien_aqui.txt -> mis_datos.txt

gwolf@lactop:/tmp$ cat >> tambien_aqui.txt
Mis datos siguen creciendo!
gwolf@lactop:/tmp$ cat mis_datos.txt
Estos son otros datos, y aunque no parezca, los quiero también.
Mis datos siguen creciendo!

gwolf@lactop:/tmp$ rm mis_datos.txt
gwolf@lactop:/tmp$ ls -l tambien_aqui.txt
lrwxrwxrwx 1 gwolf gwolf 13 2005-12-08 13:05 tambien_aqui.txt -> mis_datos.txt
gwolf@lactop:/tmp$ cat tambien_aqui.txt
cat: tambien_aqui.txt: No such file or directory
gwolf@lactop:/tmp$
```

## 2. El uso de las ligas en los sistemas de archivos en Unix

Agradecemos juntos a cp!

---

El comando de copia en Unix, `cp`, nos simplifica la creación de ligas cuando lo requiramos - Citando a `man cp` (del `cp` de GNU):

```
(...)  
-H      follow command-line symbolic links  
-l, --link  
        link files instead of copying  
-L, --dereference  
        always follow symbolic links  
-P, --no-dereference  
        never follow symbolic links  
-p      same as --preserve=mode,ownership,timestamps  
--preserve[=ATTR_LIST]  
        preserve the specified attributes (default: mode,owner-  
        ship,timestamps), if possible additional attributes: links, all  
(...)  
-s, --symbolic-link  
        make symbolic links instead of copying
```

En este inciso, estamos hablando únicamente de sistemas Unix.

En Windows no existen las ligas duras.

Windows puede participar en el esquema de respaldos que proponemos únicamente como *cliente*

Sin embargo... Todos sabemos que hablar de seguridad y Windows es por sí sólo ridículo ;-). Así que no se preocupen. Ni el mejor de los respaldos ayuda.

# Respaldos multinivel robustos y simples utilizando rsync

1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra
2. El uso de las ligas en los sistemas de archivos en Unix
- 3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)**
4. El sistema rsync
5. Implementación simple de un esquema de respaldos multinivel

### 3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)

¿Para qué?

---

- Para llevar a cabo un respaldo remoto, requerimos un método para permitir a nuestros servidores acceso al servidor de respaldos
- El acceso -en el caso de nuestra implementación- tiene que ser con privilegios de superusuario, pues de otro modo no se podrían reflejar todos los metadatos de los archivos
- El acceso con privilegios de superusuario significa que otorgamos **acceso completo, absoluto e irrestricto al sistema**, y debe ser manejado con sumo cuidado, limitándolo tanto como sea posible.
- Para acabarla de amolar, en mi Instituto no contamos con un servidor dedicado a los respaldos (y los realizamos en mi computadora personal)

### 3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)

#### Generación de nuestro par de llaves

---

En el servidor del cual crearemos respaldos:

```
root@server:~# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
30:bd:01:a0:8a:12:1f:4a:75:09:5f:bd:6c:d9:10:65
```

Importante - Y contrario a toda recomendación que hayan visto: Tenemos que crear la llave *sin contraseña*

- El respaldo va a ser automático
- Poner una contraseña en un script es peor que no tener dicha contraseña
- Como sea: Estén conscientes del riesgo!
  - Un atacante puede aprovechar esta llave para *eliminar todos nuestros respaldos*

### 3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)

Envío de la llave al servidor de respaldos

---

- Permitimos acceso como administrador en nuestro servidor de respaldos - en `/etc/ssh/sshd_config` (o `/etc/ssh/sshd_config`):

```
PermitRootLogin yes
```

- Enviamos la llave al servidor de respaldos, y comprobamos que funcione correctamente entrando - ya no debe solicitarnos contraseña.

```
root@server:~# ssh-copy-id root@respaldos
```

```
Password:
```

```
Now try logging into the machine, with "ssh 'localhost'", and check in:
```

```
  .ssh/authorized_keys
```

```
to make sure we haven't added extra keys that you weren't expecting.
```

```
root@server:~# ssh root@respaldos
```

```
Linux respaldos 2.6.14-lrespaldos #1 Sat Nov 26 02:40:47 CST 2005 i686 GNU/Linux
```

```
root@respaldos:~#
```

### 3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)

¿Y si no tengo un servidor dedicado?

---

Si no contamos con un equipo para destinar a servidor de respaldos y no nos gusta tener llaves sin contraseña, no todo está perdido - Podemos construir un entorno protegido y casi autónomo dentro de nuestro servidor - un *chroot*

- Chroot es una facilidad que provee todo sistema tipo Unix
  - Aísla a un proceso y sus hijos del resto del sistema, cambiando el directorio raíz
  - Oculta todo lo que queda fuera del directorio especificado
  - Puedo hacer una instalación de un sistema operativo completa dentro de este directorio
    - ▶ Puede ser incluso una distribución diferente del sistema anfitrión, siempre y cuando funcione con el mismo núcleo
- Dentro de mi entorno *chroot* puedo correr ssh
  - Aislado la sesión entrante del resto del sistema
  - Los usuarios del sistema huésped son independientes de los del sistema anfitrión
  - No es necesario que dé `PermitRootLogin` en mi sistema anfitrión
  - Incluso puedo correrlo a determinada hora y cerrarlo una vez recibida la conexión de los respaldos diarios

### 3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)

¿Y si no tengo un servidor dedicado? - ¡AGUAS!

---

No todo es miel sobre hojuelas... Hay un par de puntos **importantes**:

- Un chroot da una protección *razonable* al sistema anfitrión. **No es invulnerable**, y sigue siendo una posible puerta de entrada para ataques. En nuestro caso:
  - El sistema anfitrión es mi computadora personal
  - Toda la información que tengo ahí tiene respaldo
  - No guardo información confidencial (fuera de mis hábitos de navegación ;-)
- El sistema no es visible desde Internet
  - Hace falta comprometer a uno de los servidores del Instituto primero
- ...Y aún así, estoy tomando demasiados riesgos. ¿Por qué no mejor enchufo una máquina viejita?
  - Es agradable tener 400GB en el escritorio ;-)

## Es igual

- En Windows hay varios clientes gráficos y amigables de ssh
  - Incluyendo varios de intercambio de archivos (scp, sftp)
- Pero lo que mejor sirve para nuestros fines, existe también un port basado en el OpenSSH que todos queremos y amamos (<http://sshwindows.sourceforge.net/>), y hay varias otras alternativas disponibles (<http://www.openssh.com/windows.html>)
- Todo lo mencionado en esta sección aplica exactamente igual en Windows

## Respaldos multinivel robustos y simples utilizando rsync

1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra
2. El uso de las ligas en los sistemas de archivos en Unix
3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)
- 4. El sistema rsync**
5. Implementación simple de un esquema de respaldos multinivel

Rsync es un protocolo y un programa para la sincronización de depósitos de archivos binarios

- **Del manual de rsync:**

- o support for copying links, devices, owners, groups, and permissions
- o exclude and exclude-from options similar to GNU tar
- o a CVS exclude mode for ignoring the same files that CVS would ignore
- o can use any transparent remote shell, including ssh or rsh
- o does not require root privileges
- o pipelining of file transfers to minimize latency costs
- o support for anonymous or authenticated rsync daemons (ideal for mirroring)

No entro en detalles - El manual de rsync es muy claro y completo.

Únicamente para asegurar: rsync puede operar a través de su propio demonio o a través de un shell como rsh (malo, muy malo) o ssh (bueno). No usen el demonio de rsync, usen ssh.

## 2. El sistema rsync

¿Cómo lo invoco?

---

```
root@servidor:~# rsync -q -a -l -H -o -g -D -t -e 'ssh -p 22022'  
--delete-excluded /bin /boot /dev /etc /home /lib /opt /root /sbin /srv /usr  
/var --exclude /var/spool/squid root@respaldos:/respaldo/servidor
```

# ¡¿Huh?!

- -q: Operación silenciosa (sólo muestra errores)
- -a: Modo de archivo/espejo
- -l: Mantiene las ligas simbólicas como tales
- -H: Mantiene las ligas duras como tales
- -o: Conserva la información del dueño de cada archivo
- -g: Conserva la información del grupo de cada archivo
- -D: Respaldar los árboles de dispositivos Unix
- -e 'ssh -p 22022': Utiliza un shell remoto en vez del protocolo rsync
- <directorios>: Los directorios a respaldar
- -exclude <directorios>: Los directorios a excluir del respaldo (p.ej. áreas temporales, caché, etc.)
- --delete-excluded: Si existe en el destino un directorio marcado como excluido, eliminarlo
- root@respaldos:/respaldo/servidor: Usuario, servidor y ruta destino para enviar los datos

## 2. El sistema rsync

¡Ahorre ancho de banda! ¡Use rsync!

---

Rsync es un protocolo muy eficiente para comparar grandes conjuntos de datos, evitando enviar repeticiones

- Los archivos que no se han modificado no se tienen que reenviar
- Los archivos grandes que sí han sido modificados se separan en bloques, y sólo se envían los bloques modificados
  - Los demás bloques son comprobados por un algoritmo de checksum de 128 bits

No quieres que tu servidor de respaldos sea Windows.  
No, en serio. No quieres.

Ok, ¿necesitas razones? La principal es porque Windows no puede manejar ligas duras - y a continuación veremos su importancia... Sin embargo, sí, existe rsync como cliente y como servidor para Windows

- La mayor parte de los documentos en la red apuntan a que lo mejor es instalar rsync utilizando Cygwin
  - Cygwin es una implementación del API completo de Unix
  - Es muy grande, sin embargo, y es recomendable evitarlo si no es indispensable.
- <http://www.gaztronics.net/rsync.php> menciona una implementación nativa

## Respaldos multinivel robustos y simples utilizando rsync

1. Los sistemas de respaldos tradicionales - Puntos a favor y en contra
2. El uso de las ligas en los sistemas de archivos en Unix
3. Autenticación ssh por intercambio de llaves (y un par de tips limitantes)
4. El sistema rsync
5. Implementación simple de un esquema de respaldos multinivel

Vimos ya mucho rollo muy disperso hasta este momento...

**Basta de darle vueltas  
¡Vamos a integrar!**

## 5. Implementación simple de un esquema de respaldos multinivel

### Esquema general

---

- Respaldo periódico utilizando rsync
- Comunicación entre los servidores y el equipo de respaldos vía ssh con intercambio de llaves
- Antes de iniciar el respaldo, copiamos el respaldo actual en el servidor de respaldos
  - La copia se hace con un simple `cp -a1`, evitando ocupar espacio adicional con los archivos que no han sido modificados
  - El único espacio adicional que se ocupa es la duplicación del árbol de directorio (y, claro, las diferencias entre los varios árboles)

## 5. Implementación simple de un esquema de respaldos multinivel

### Recuperando un archivo del respaldo

---

- ¿Cómo se llamaba el archivo?

```
root@servidor:~# scp respaldos:/respaldo/servidor/home/usuario/archivo /home/usuario/archivo
```

- No recuerdo el nombre, pero era un PDF

```
root@servidor:~# ssh respaldos
```

```
root@respaldos:~# cd /respaldo/servidor/home/usuario
```

```
root@respaldos:~# ls *.pdf
```

```
(...)
```

```
root@respaldos:~# logout
```

```
root@servidor:~# scp respaldos:/respaldo/servidor/home/usuario/archivo.pdf /home/usuario/
```

- No lo reconozco, pero hablaba acerca del desarrollo del agro...

```
root@servidor:~# ssh respaldos
```

```
root@respaldos:~# cd /respaldo/servidor/home/usuario
```

```
root@respaldos:~# grep -i agro *.pdf
```

```
root@respaldos:~# logout
```

```
root@servidor:~# scp respaldos:/respaldo/servidor/home/usuario/archivo.pdf /home/usuario/
```

En fin, su uso es como el uso de un sistema de archivos normal Unix.  
Sin embargo...

## 5. Implementación simple de un esquema de respaldos multinivel

Auxiliando al administrador del sistema

---

- ¿Qué configuraciones cambié de ayer a hoy?

```
root@respaldos:~# cd /respaldo/  
root@respaldos:/respaldo# diff -ur servidor.1/etc servidor.2/etc
```

- ¡Un exploit en /var/tmp/.bash\_history! ¡Tengo que encontrar la puerta que utilizaron en la bitácora! ¿Hace cuántos días habrán entrado?

```
root@respaldos:/respaldo# ls -dl servidor*/var/tmp/.bash_history
```

En fin... Como todo en la administración: Tu imaginación es el límite :)

## 5. Implementación simple de un esquema de respaldos multinivel

El código

---

(... ¡Vámonos para Emacs! )

# ¿Dudas?

gwolf@gwolf.org

[http://www.gwolf.org/seguridad/respaldos\\_con\\_rsync/](http://www.gwolf.org/seguridad/respaldos_con_rsync/)

Muchas gracias a Mike Rubel, pues la idea e implementación original es suya. Su artículo:

[http://www.mikerubel.org/computers/rsync\\_snapshots/](http://www.mikerubel.org/computers/rsync_snapshots/)